PR #41383 完整报告

vllm-project/vllm

[Nixl][PD] Lease renewal TTL KV blocks on P

合并时间：2026-05-11 17:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41383

执行摘要

- 一句话：新增 NIXL 心跳租约续期，优化 KV 块保留时间
- 推荐动作：值得所有关注分布式 KV 传输的开发者精读。设计上展示了典型的租约续期模式，接口抽象和配置简化决策值得借鉴。建议重点阅读 scheduler.py 的 on_new_request/_stop_heartbeat 和 worker.py 的 _ensure_handshake/_send_heartbeats，体会如何在分布式组件间实现轻量级心跳协议。

功能与动机

原有的 VLLM_NIXL_ABORT_REQUEST_TIMEOUT 单一超时过于简单，当 D 崩溃时会导致 P 端长时间持有远程请求的 KV 块，造成资源浪费。需要一套动态 TTL " 租约续期 " 系统，最小化块在 P 端的滞留时间，同时让 D 能够在等待队列拥堵时延长 P 端块的租约。设计文档详见 Google Docs 链接。

实现拆解

1. 元数据扩展：新增 HeartbeatInfo dataclass，在 NixlConnectorMetadata 中添加 heartbeat_by_engine 字典，用于调度器传递心跳信息给 Worker。
2. 调度器心跳追踪：在 NixlConnectorScheduler 中增加 _heartbeat_by_engine、_heartbeat_req_engine、_last_heartbeat_time 数据结构，实现 on_new_request 方法在请求入队时注册远程引擎关联的心跳信息，_stop_heartbeat 方法在请求发送完成或终止时移除对应条目。心跳发送节流通过比较 _last_heartbeat_time 与当前时间控制。
3. Worker 心跳收发：新增 _send_heartbeats 方法，在 start_load_kv 时从元数据中提取心跳信息并向每个远程引擎发送心跳消息（ZMQ）；_handle_heartbeat 在 P 端接收心跳后延长 _reqs_to_send 中对应请求的过期时间（time.perf_counter() + self._lease_extension）。
4. 接口整合：KVConnector 基类增加 on_new_request 和 update_connector_output 方法，分别用于通知连接器新请求入队和处理 Worker 输出（停止心跳）。NixlConnector 和 MultiConnector 转发调用。
5. 配置简化：将原先的三个配置项（heartbeat_interval、heartbeat_lease_extension、initial_kv_lease）合并为单一 kv_lease_duration，内部派生 _heartbeat_interval = kv_lease_duration // 6，_lease_extension = kv_lease_duration * 2 // 3。同时新增 decoder_kv_blocks_ttl 配置用于双向传输中 D 端块的固定 TTL。

6. 测试覆盖：新增 tests/v1/kv_connector/unit/test_nixl_heartbeat.py，包含 5 个单元测试验证心跳跟踪、分组、节流、停止路径。同时修改测试工具函数 make_nixl_scheduler 支持心跳参数。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/nixl/scheduler.py（模块 NIXL 调度器；类别 source；类型 core-logic；符号 on_new_request, _stop_heartbeat, update_connector_output）：核心逻辑变更，实现调度器端的心跳追踪和停止逻辑
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/worker.py（模块 NIXL Worker；类别 source；类型 dependency-wiring；符号 _background_nixl_handshake, _ensure_handshake, done_callback, _handle_heartbeat）：Worker 端实现心跳收发和租约扩展，重构 handshake 逻辑
- tests/v1/kv_connector/unit/test_nixl_heartbeat.py（模块 心跳测试；类别 test；类型 test-coverage；符号 _sched, _req, _worker_stub, test_on_new_request_tracks_and_groups）：新增单元测试覆盖心跳功能，验证追踪、节流、停止路径
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/metadata.py（模块 元数据；类别 source；类型 core-logic；符号 HeartbeatInfo）：新增 HeartbeatInfo 数据类和版本更新
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/connector.py（模块 连接器桥接；类别 source；类型 core-logic；符号 on_new_request, update_connector_output）：连接器桥接调度器和 Worker，新增 on_new_request 和 update_connector_output 方法
- vllm/distributed/kv_transfer/kv_connector/v1/base.py（模块 基类接口；类别 source；类型 core-logic；符号 on_new_request）：基类接口扩展，增加 on_new_request 默认实现

关键符号：on_new_request, _stop_heartbeat, update_connector_output, _handle_heartbeat, _ensure_handshake, _send_heartbeats

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/nixl/worker.py

Worker 端实现心跳收发和租约扩展，重构 handshake 逻辑

```
    def _ensure_handshake(
        self,
        engine_id: EngineId,
        host: str,
        port: int,
        tp_size: int,
    ) -> Future[dict[int, str]] | None:
        """
        确保与远程引擎的握手已完成或正在执行。

        如果握手已成功完成则直接返回 ``None``，如果握手正在进行则返回 ``Future``，
        如果尚未开始则启动并返回 ``Future``。
        调用方可以在返回的 ``Future`` 上附加 per-request 回调。
        """
```

```
        with self._handshake_lock:
            # 检查是否已有成功的远程代理连接
            if engine_id in self._remote_agents:
                return None # 握手已完成
            # 检查是否有正在进行的握手
            fut = self._handshake_futures.get(engine_id)
            if fut is not None:
                return fut # 等待中的握手
            # 提交新的握手任务
            fut = self._handshake_initiation_executor.submit(
                self._nixl_handshake,
                host,
                port,
                tp_size,
                engine_id,
            )
            self._handshake_futures[engine_id] = fut

            def done_callback(f: Future[dict[int, str]], eid=engine_id):
                """握手完成后的清理与错误日志回调"""
                with self._handshake_lock:
                    del self._handshake_futures[eid]
                try:
                    f.result() # 触发异常如果失败
                except Exception as e:
                    # 记录握手失败日志
                    logger.error('Handshake failed for engine %s: %s', eid, e)

            fut.add_done_callback(done_callback)
            return fut
```

tests/v1/kv_connector/unit/test_nixl_heartbeat.py

新增单元测试覆盖心跳功能，验证追踪、节流、停止路径

```
def test_on_new_request_tracks_and_groups():
    """添加两个请求到同一引擎，一个到另一引擎；验证分组正确"""
    s = _sched() # 创建带心跳支持的调度器
    s.on_new_request(_req(1)) # 添加请求 1（默认引擎 _ENGINE_A）
    s.on_new_request(_req(2)) # 添加请求 2（同一引擎）

    # 验证同一引擎下两个请求的心跳 ID 集合
    assert s._heartbeat_by_engine[_ENGINE_A].req_ids == {'prefill-1', 'prefill-2'}
    info = s._heartbeat_by_engine[_ENGINE_A]
    # 验证远程主机、端口、TP 大小与模拟数据一致
    assert (info.host, info.port, info.tp_size) == ('my-host', 1234, 1)
    # 验证反向映射表
    assert s._heartbeat_req_engine['id-1'] == (_ENGINE_A, 'prefill-1')

    # 添加不同引擎的请求
```

```
r3 = _req(3)
r3.kv_transfer_params['remote_engine_id'] = 'engine-b'
s.on_new_request(r3)
# 此时应有两个引擎条目
assert len(s._heartbeat_by_engine) == 2
```

评论区精华

主要讨论集中在接口设计和配置简化。

- @markmc 建议将三个配置项合并为单一 kv_lease_duration，避免用户暴露过多细节。作者采纳，最终接口仅暴露 kv_lease_duration。
- @orozery 提议调度器只需在 add_request 时通知连接器，避免复杂的 ObservableQueue 抽象。经过讨论，作者放弃 ObservableRequestQueue，改用直接的 connector.on_new_request 回调。
- @ivanium 质疑 ObservableRequestQueue 的放置位置，建议移到连接器内部，最终决定保持简单。
- @markmc 指出 _send_heartbeats 访问 self._remote_agents 时未持有 _handshake_lock，存在数据竞争。作者修复。
- 关于双向传输场景，@markmc 认为 D 端块的 TTL 应独立配置且与心跳无关，最终引入 decoder_kv_blocks_ttl 配置，默认 480 秒。
- 简化配置项数量 (design): 采纳，最终只暴露 kv_lease_duration 给用户，内部按固定比例派生其他值。
- 远程引擎追踪优化 (performance): 采纳，修改为仅在 remote_engine_id 不在字典中时创建新 HeartbeatInfo。
- 心跳竞态条件 (correctness): 作者认为问题不大，未做额外修改。markmc 也认为不大可能覆盖更晚的过期时间。状态 unresolved。
- ObservableRequestQueue 设计争议 (design): 移除 ObservableRequestQueue，改用直接的 connector.on_new_request 回调。
- 双向传输模式 TTL 独立配置 (design): 采纳，新增 decoder_kv_blocks_ttl 配置，默认为 480 秒。
- 锁保护 remote_agents 访问 (security): 作者修复，添加 with self._handshake_lock 保护。

风险与影响

- 风险：
  1. 线程安全：_send_heartbeats 和 _ensure_handshake 对 _remote_agents 的访问虽已加锁，但其他路径（如 _handle_heartbeat 更新 _reqs_to_send）未显式加锁，可能在高并发下出现竞态。
  2. 配置兼容性：废弃 VLLM_NIXL_ABORT_REQUEST_TIMEOUT 环境变量，用户需迁移到新的 kv_lease_duration 配置，否则将使用默认值 30 秒。
  3. 双向传输风险：P 端不发送心跳，D 端依赖固定 TTL，若 decoder_kv_blocks_ttl 设置过短，在 D 端请求处理延迟时可能导致块提前释放。

4. 性能风险：心跳消息增加网络开销，但节流机制（每 kv_lease_duration/6 秒一次）将其控制在可接受范围。 - 影响：影响范围限于使用 NIXL KV 连接器的分布式推理场景。主要好处是降低了 P 端块的无谓滞留，提高显存利用率，并允许 D 端通过心跳主动延长块租约，减少因拥塞导致的块提前释放。用户需要为 kv_lease_duration 设置合适的值。测试发现对正常吞吐无显著影响。团队需维护新增的 4 个配置项和对应的测试。 - 风险标记：核心路径变更，线程安全，双向传输兼容性，配置废弃

关联脉络

- PR #40364 [KV Connector][NIXL][Bugfix] Fix NIXL handshake failures not honoring kv_load_failure_policy: 修改相同的 NIXL worker 和 scheduler 文件，涉及握手失败处理，本 PR 依赖其修复后的基础机制。
- PR #41806 fix nixl side-channel host selection: 也修改了 NIXL 相关 worker/scheduler，本 PR 的心跳通信可能受 side-channel 配置影响。